Test Plan for Chatbot Development for Medi-Kit Curtin Health Project

Team 102_CH

This document outlines the test plan for the chatbot developed using the Gemini API and Flutter. The goal is to ensure the chatbot meets the client's expectations in terms of functionality, performance, and user experience.

Notes and terms:

- The test plan details are open for modification.
- **To be determined:** details required clients' clarification during the next meeting.
- **To be discussed:** details to be discussed between team members.

1. Test Objectives

- Ensure the chatbot functions correctly as client expectations.
- Validate the integration between the Gemini API and the Flutter app.
- Confirm that the chatbot provides accurate and relevant responses to use cases.
- Test the user interface for usability and responsiveness.

2. Test Scope

- **Functional Testing:** Cover all features and functionalities of the chatbot.
- **API Testing:** Focus on the Gemini API's reliability and accuracy.
- **UI/UX Testing:** Assess the user interface and experience on specified devices.
- **Performance Testing:** Evaluate the chatbot's efficiency.
- **Security Testing:** Test the chatbot for potential security vulnerabilities.

3. Test Environment

- **Development Environment:** Use the latest stable versions of Flutter and the Gemini API. Current potential testing environments are:
    - Slack
    - Local computer
- **Devices:** Test on a variety of Android and iOS devices using team members' devices, including different screen sizes. Potentially test on a client's similar device.
- **Tools:**
    - Testing Framework: Flutter testing framework, any UI/UX testing tool like Flutter Driver, pytest for backend unit testing
    - Manual Testing with team members and clients.

4. Test Cases

4.1 Functional Testing

Use cases

Table 1					
ID	**User story**	**Given**	**When**	**Then**	**Case type**
1	As a general practitioner/ nurse, I try to ask for other information that might not be related to chatbot function.	I am not all familiar with the chatbot policies and restrictions.	I accidentally ask unrelated or unallowed questions.	The chatbot refused to answer due to restrictions.	**2**
2	The general practitioner ask about the hypertension stage 2 blood pressure information	I need information about blood pressure at hypertension stage 2, which should be context related to my clinic place.	I just took a blood pressure test for the patient and need to verify potential hypertension status.	I got information about the blood pressure in hypertension stage 2.	**2**

Potential case types:

- Case 1: Basic User Interaction
    - Description: Test the chatbot's response to a simple greeting (e.g., "Hello").
    - Expected Result: The chatbot should reply with a greeting.
- Case 2: Complex Query Handling
    - Description: Ask the chatbot a complex question related to its purpose.
    - Expected Result: The chatbot should provide an accurate and relevant response.
- Case 3: Error Handling
    - Description: Input an ambiguous or nonsensical query.
    - Expected Result: The chatbot should handle the error gracefully, possibly with a prompt for clarification.

4.2 API Testing

- **Case 1: API Response Time**
    - **Description:** Measure the time taken for the chatbot to receive a response from the Gemini API.
    - **Expected Result:** Response time should be within acceptable limits (e.g., <3 seconds).
- **Case 2: API Failure Handling**
    - **Description:** Simulate a failure in the Gemini API.

- **Expected Result:** The chatbot should display an appropriate error message and handle the failure without crashing.

4.3 UI/UX Testing

Case 1: Client-Specified UI Consistency

- **Objective:** Ensure the chatbot's UI meets the client's design requirements.
- **Expected Result:** UI should be consistent, aesthetically pleasing, and aligned with the client's Figma document.

Case 2: Navigation and Usability

- **Objective:** Test the ease of navigation and overall usability.
- **Expected Result:** Users should be able to interact with the chatbot without confusion or difficulty. Workflow as shown in the Figma document provided by clients.

4.4 Performance Testing Show the chatbot to the client (not in project scope)

- **Case 1: Load Testing (If the chatbot will be performed in multiple MediKits)**
  - **Description:** Simulate multiple users interacting with the chatbot simultaneously.
  - **Expected Result:** The chatbot should handle the load without significant performance degradation.

4.5 Security Testing (required user login information - not in project scope)

- **Case 1: Data Privacy (if the chatbot is allowed to store patients' information)**
  - **Description:** Test how the chatbot handles sensitive information.
  - **Expected Result:** Sensitive data should be protected, and the chatbot should not expose any private information.

5. Test Execution

- **Timeline:** Provide a schedule for when each type of testing will be conducted.
  - Internal: component test (backend/frontend) - integration test - deployment test
  - Timeframe: executed frequently via CI/CD workflows and after merging branches
    - Week 6: Basic testing via video recording or screenshot
    - Week 8: Component testing (backend, frontend), Integration testing
    - Week 9/10: Deployment testing
    - Before handover: via handover documentation and demo
- **Testers:** Assign testers to each type of testing based on their expertise.
  - Backend
  - Frontend
  - Integration

- **Format (To be discussed further)**
    - **Updated:**
        - **AC/AT:** As description in Jira Tickets
        - (if applicable) local testing for Pull Requests (PRs) before approving
        - **Feedback:** Comment to corresponding Jira Tickets or PRs
    - For function testing, the details of each test case will be formatted as follows (see Table 2 for example from Use case 1). Otherwise, Testmo can be used for its convenience when working with Jira.
    - For UI/UX testing, the current method will be recording video of the workflow when using chatbot, then asking for feedback from clients.

Table 2 (from Lecture 5 - Testing)	
Test Type: Functional	**Execution Type:** Manual
Objective: The chatbot gives appropriate responses about suggestions for general practitioners/nurses on what procedure they should do to patients.	
Setup: The app is used together with a well-equipped MediKit device. The practitioner is able to perform the suggested diagnosis method.	
Note and preconditions:	
Steps: 1. The user asks the bot: “Hi Ed, Darlene has a fever and feels dizzy. What should I do next?” 2. Verify that the responses given by the bot are appropriate and fit to context.	
Time constraint: - Minimum time (To be determined/discussed) - Maximum time (To be determined/discussed)	

6. Test Reporting

- **Bug Reporting:** Document any bugs found during testing, including steps to reproduce, severity, and screenshots.
- **Test Summary:** Provide a summary of the test results, including pass/fail rates for each test case.
- Documents are saved into Jira/Confluence for reference.

7. Acceptance Criteria

- All critical and high-severity bugs must be fixed.
- Functional requirements must be met with at least 90% test coverage.

- Performance metrics must meet the predefined benchmarks.

Reference:

[An Ultimate Guide To Chatbot Testing Checklist | KiwiQA Blog]
Lecture 5 - 19/8/2024 - Testing 1, IT Project COMP30022, University of Melbourne